Introduction to Version Control with Git

What is git

- A version control system
- A command line tool for keeping track of project changes
- A contemptible person
- A tool affectionately named for its creator.
- Awesome

what is Version Control

"The task of keeping a software system consisting of many versions and configurations well organized" - Google Definitions

Why use version control

- Makes it easier to share code (emailing files doesn't count)
- Provides a history of changes
- A spilled coffee is less likely to cost you all of your work

some other popular VCS's

- SVN
- CVS
- Mercurial

Repositories

Repositories (aka repos) are miniature filesystems tracked by git. When you work with git you keep your changes in a repository.

Local vs remote repos

- Local repository: on your computer; only you (usually) have access to it. Almost always a copy of a remote repo.
- Remote repository: A repository on another machine (a server or another team member)

Changes are usually made by each team member to their local repo and then copied to the remote repo (usually a shared repo for the whole team or project) when ready.

Git commands

Set up a local repo

- `git init` - creates a new local repo in the current directory
- `git clone remoterepo` - Copies remoterepo (another git repo) to a folder in the current directory and creates a local repo in that folder

---

Git status

Tells you the state of your local repo. Will list files that have been changed and any changes staged (waiting) for commit (saving to the repo)

```
$ git status
On branch gh-pages
Your branch is up-to-date with 'origin/gh-pages'.
nothing to commit, working directory clean
```

Adding to your local repo

Files created in a directory with a git repo are not automatically added.
Stage (prepare) files to be added to a repo with `git add filename`
Commit staged changes with `git commit`

```
$ touch newfile
$ git add newfile
$ git status
On branch gh-pages
Your branch is up-to-date with 'origin/gh-pages'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   newfile
```

Collaboration

Make changes and test your code on a local repo. Then share the finished result on a shared remote repo.

Pulling from remote

When a remote repo has new changes, bring them into your local repo with `git pull`

Pushing to remote

Cloned repos automatically have a remote named `origin`. Submit your commits using `push` and the remote and branch name.
Eg: `git push origin featurebranch` (it's good practice not to push untested code to master)

Branching

Branches are a way to store work in progress, such as new features or untested code.

---

create branch, checkout, and merge

- `git branch mybranch` - creates new branch called mybranch
- `git checkout mybranch` - switches to mybranch. Any new changes committed become part of mybranch

---

Merging and merge conflicts

- `git merge anotherbranch` - Adds commits from anotherbranch to the current branch. Can cause conflicts.

Resolving conflicts can be time consuming, so it's best to coordinate efforts and always work from the latest code.

Listing branches

- `git branch` - list local branches, * indicates current branch
- `git branch -a` - lists all branches, including remote branches

Where do remote repos live?

- Someone else's machine
- A server that hosts the project code
- Online services like GitHub or Bit Bucket
- Somewhere else on your machine (this gets messy)

GitHub

What is github?

A website that provides free public (and paid private) git repo hosting and some really nice tools.

GitHub is possibly the most popular place to host open source software

---

Forking repositories (similar to cloning)

Github repos have a "fork" button. Clicking this creates a clone of the repo in your own account.
Effectively a clone on the same (maybe?) server.
When you fork a repo, you can make any changes to your fork without impacting the original repo.

---

cloning your fork

Your fork on GitHub is a repo just like any other. Clone it to do work locally.

---

pull requests

Made some awesome changes to your fork of someone else's project and want them to include it in their code?
You can issue a Pull Request (a button on your GitHub repo) asking them to pull your code into their repo.

---

upstream remotes

The project that a repo was forked from is often referred to as the "upstream" repo. Adding upstream repos as remotes in your local repo can be helpful for applying patches or testing upstream changes.

Relax - lots of practice and good guides out there

- Github Guides
- Code School Try Git
- Codecademy Learn Git

Recap: basic commands

- `git init` - create a new repo here
- `git clone` - copy a remote repo into a directory here
- `git status` - check the status of my repo
- `git add` - add changes to staging
- `git commit` - save staged changes
- `git branch branchname` - create a branch named branchname
- `git checkout branchname` - switch to branchname
- `git merge otherbranch` - add commits from otherbranch to this branch

Catastrophic recovery

A bit of sage advice